# Developing a Matlab Controller for Niryo Ned Robot

Praneel Chand [1]

[1] Centre for Engineering and Industrial Design
Waikato Institute of Technology
Hamilton, New Zealand
Email: praneelchand10@yahoo.co.nz [1]

2022 1st International Conference on Technology Innovation and Its Applications (ICTIIA) | 978-1-6654-8826-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICTIIA54654.2022.9935911

***Abstract*— This paper presents the development of a novel Matlab controller for the Niryo Ned Robot. The Ned is an open-source technology robotic arm for education, research, and vocational training. The Ned robot comes equipped with Niryo Studio that can be used to configure and implement basic programs. Niryo Studio is well suited for straightforward education use. Currently, controlling the Ned robot via Matlab is limited as there are no functional controllers for this. A Matlab controller has the advantage that it can be used for research and development by leveraging various toolboxes for machine learning, control, and image processing. The developed Matlab controller is tested on a real robot. Four target coordinate locations representing the edges of the robot's workspace are evaluated. Future work will include the development of additional control features in the Matlab controller.**

***Keywords*— robot arm, control, Matlab, Robot operating system (ROS), Niryo Ned robot**

## I. Introduction

Robotic arms have become an integral part of modern society with numerous applications in industry [1] [2]. Advances in technology have facilitated the development of robotics as a practical subject that is now taught in schools [3] [4] [5]. Many robotic arms are now termed as collaborative robots (cobots) as they become capable of direction human interaction within a shared workspace [6]. Robotics and automation education and research provide an opportunity to create change and develop solutions to problems faced in society. While the affordability of robotic arms has improved over the years [7], they still require considerable investment regardless of use in industry or in educational institutions.

Industrial and commercial robotic arms are generally most expensive. The median price of these robots in 2021 was US$ 22,600 [7]. Some representative manufacturers robotic arms with small payload capabilities for light assembly tasks include ABB's IRB 120 [8], Omron's Viper 650 [9], Kuka's KR 4 Agilus [10], and Universal Robots' UR3e [11]. These robots typically come with proprietary licensed software and a fixed set of customization options. They may be suitable for classroom training, but the lack of customization may be a hindrance for research applications.

Small scale versions of industrial and commercial robotic arms are available for education and research. Typically, these robots cost much less and support open-source technologies. This facilitates the access of robotic arms to the wider society. Some representative examples of robots with open-source technologies are summarized in Table I. These robots support a range of programming environments such as Matlab, Python, and Robot Operating System (ROS), in addition to the manufacturer supplied software suite. These robots cost US$ 1326 to US$ 3299 and are much more affordable than industrial and commercial robotic arms.

Matlab is a popular software used by engineers and scientists to solve complex design problems. It has a myriad of toolboxes that can be accessed to prototype solutions. Matlab can be used for research and development of advanced robotic control applications by leveraging various toolboxes for machine learning, control, and image processing. Based on this primary advantage, the availability of a local distributor, and detailed product documentation, the Niryo Ned robot has been selected for use. A key challenge for getting an ROS based robot to work with Matlab is setting up communication between the Linux based controller and Matlab's ROS toolbox. Robot manufacturers need to provide library files so that ROS toolbox commands can be generated for controlling a robot.

Hence, this paper details the process of developing a Matlab controller for the Niryo Ned. First, the process of setting up communication between the Niryo Ned robot and a Windows computer running Matlab is outlined. Following this, the design and development of the Matlab controller is presented. Experiments are conducted to evaluate the control of the robot.

## II. Methods

### A. Conceptual Framework and Proposed Robot Task

The conceptual framework of the Niryo Ned robotic arm control system is shown in Fig. 1. Ned uses ROS to interface between the robot hardware and higher-level control programs.

ROS provides services and topics which can be published and subscribed to for facilitating low-level device control. Higher-level control is achieved via a laptop computer running software such as Niryo Studio or Matlab. The robot's Raspberry Pi 4 controller has ROS, Python wrappers, and communication interfaces installed on it.

TABLE I. COMPARISON OF THREE OPEN-SOURCE ROBOTIC ARMS

| **Criteria** | **Niryo Ned [12]** | **WLKATA Mirobot [13]** | **Elephant MyCobot-Pi [14]** |
|---|---|---|---|
| Parameters | cobot<br>6 DoF<br>300g payload | cobot<br>6 DoF<br>150g payload | cobot<br>6 DoF<br>250g payload |
| Controller | Raspberry Pi<br>Ubuntu 18.04 and ROS | Not specified<br>Not specified | Raspberry Pi<br>Ubuntu and ROS |
| Software | Niryo Studio<br>Blockly programming | WLKATA Studio<br>Blockly programming | myStudio, RoboFlow, myblockly<br>Blockly programming |
| Matlab Support | Yes<br>Wi-Fi | Yes<br>USB serial | No<br>NA |
| Cost (US$) | 3299 | 1680 | 1326 |

The proposed Matlab based task for the robot is an object sorting system as shown in Fig. 2. An overhead camera will be used recognize and locate parts within a pre-defined workspace. Matlab's image acquisition toolbox and various interactive apps and algorithms for machine learning will be used to develop the control system.

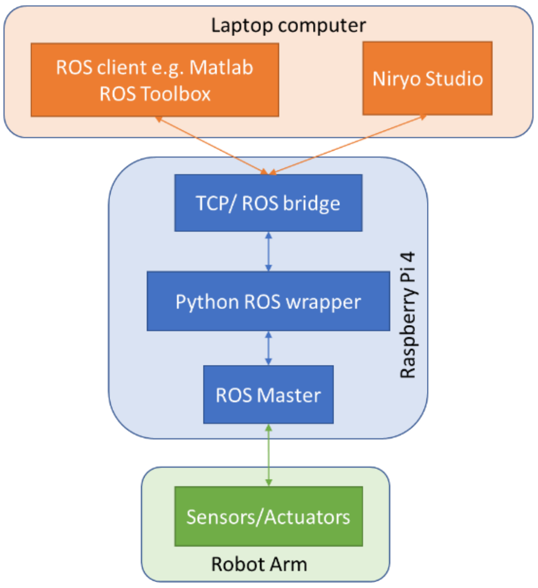


Fig. 1. Conceptual framework of Niryo Ned robotic arm control system.

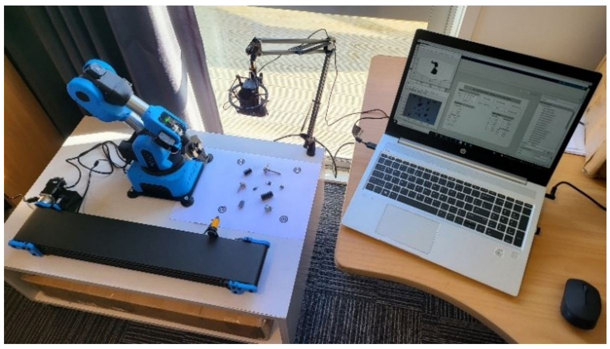
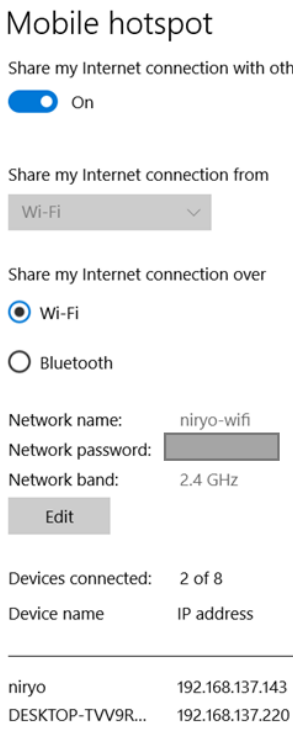


Fig. 2. Proposed robot task – object sorting system.

## *B. Setting up Communication between Raspberry Pi and Windows 10*

The most important part of getting the Ned robot to work with Matlab in Windows 10 is setting up Wi-Fi communication and assigning IP addresses. Matlab's ROS toolbox can only communicate with the Ned over Wi-Fi. Wired Ethernet is unsupported. Initially, the robot is setup to work in Hotspot mode. Through Niryo Studio, a connection can be made to the robot and the Wi-Fi settings can be changed [15]. In a home networking environment, it is relatively easy to setup Wi-Fi by obtaining the relevant network information from the router and assigning an unused IP address. In a work environment where DHCP is often used, the easiest method is to setup a mobile hotspot network connection from a Windows 10 laptop [16]. This method is employed in this paper (Fig. 3).



Mobile hotspot

Share my Internet connection with oth

On

Share my Internet connection from

Wi-Fi

Share my Internet connection over

(●) Wi-Fi

( ) Bluetooth

| | |
|---|---|
| Network name: | niryo-wifi |
| Network password: | |
| Network band: | 2.4 GHz |

Edit

Devices connected: 2 of 8

| Device name | IP address |
|---|---|
| niryo | 192.168.137.143 |
| DESKTOP-TVV9R... | 192.168.137.220 |

Fig. 3. Mobile hotspot setting and robot and laptop IPs.

Next, it is a requirement to restart the robot program in the current terminal on the Raspberry Pi to support Matlab ROS Toolbox connections. This requires an SSH connection and modification of the Linux .bashrc file. The additions to the .bashrc file are shown in Fig. 4. Recent builds of Windows 10 support SSH connection via the Windows command prompt. A connection can be made by typing `ssh niryo@<ip_address>`.

```
sudo service niryo_robot_ros stop
roslaunch niryo_robot_bringup niryo_ned_robot.launch
export ROS_MASTER_URI=http://192.168.137.143:11311
export ROS_IP=192.168.137.143
```

Fig. 4. Additions to the .bashrc file for Windows 10 Matlab support.

## C. *Matlab R2021a Windows 10 ROS Toolbox Setup*

The next important phase is setting up Matlab and the ROS Toolbox for Ned. Details are available at [17] and the key steps are summarized below:

- Install the Matlab ROS Toolbox add-on from Mathworks.
- Install Cmake version 3.19.8.
- Install Python version 2.7.18 (ROS Melodic requirement) [18].
- Download the ned_ros package from Github.
- Generate the Ned messages and services from the ROS package in Matlab.
- Within Matlab, configure communication between the Ned and Matlab.

After setting up the ROS Toolbox for Ned, some basic tutorials for the robot can be programmed and tested [19]. Ensuring the Windows clock time is synchronized when executing motion commands using the ROS Toolbox is vital.

## D. *Matlab R2021a Windows 10 Controller*

The Windows 10 controller provides a simple graphical user interface (GUI) to interact with basic functions of the robot arm. It builds and enhances some of some of the basic code in the tutorials to update the physical robot's position and simulate it. Inspired by the layout of the Niryo Studio application, it displays the cartesian position of the robot and the joint angles. It also supports IP address changes and toggling the learning mode.

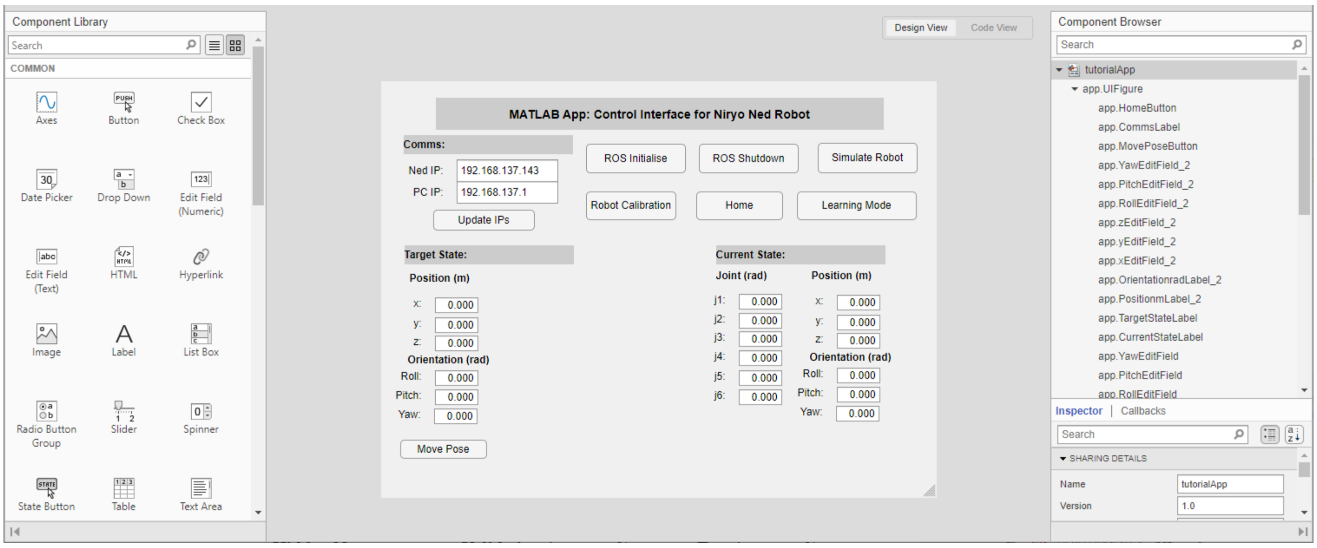
Various callback functions representing modular actions are executed when buttons are pressed. Multiple callback functions can be executed in a predefined sequence by programming customized buttons. For example, the '*Move Pose*' callback function (Fig. 5) also calls the '*Simulate Robot*' callback function (Fig. 6) to display joint angles and simulate the robot's position after movement. The '*ROS Initialise*' and '*ROS Shutdown*' buttons connect and disconnect the robot, respectively. An overview of the layout of the Matlab controller is shown in Fig. 7 and Fig. 8.

```
Create inverse kinematic client to compute joint angles
Read target pose from controller GUI
Compute inverse kinematics
Set forward kinematic controller data (joint angles and parameters)
Send forward kinematic controller data to robot
Update simulated position of robot
```

Fig. 5. Move Pose function pseudocode.

```
Read robot joint states (angles)
Compute robot configuration solution
Plot configuration solution
Update controller GUI joint angles
Update controller GUI position (x,y,z)
Update controller GUI roll, pitch, and yaw (RPY)
```

Fig. 6. Simulate Robot function pseudocode.



Ned IP: 192.168.137.143
PC IP: 192.168.137.1

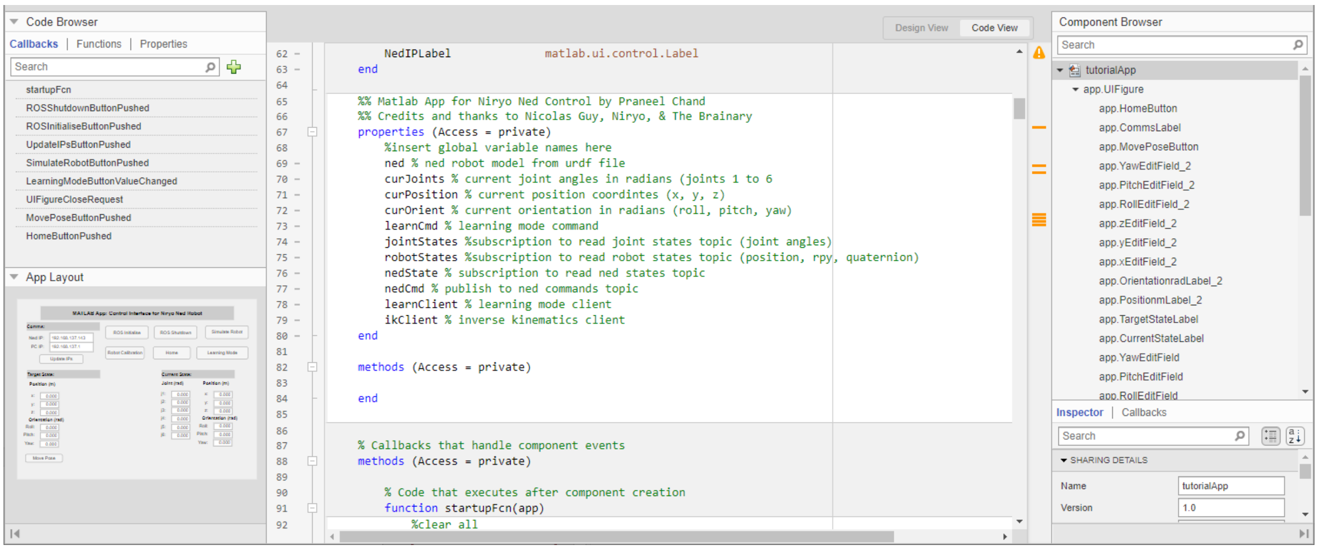
Target State: x: 0.000, y: 0.000, z: 0.000; Roll: 0.000, Pitch: 0.000, Yaw: 0.000

Current State: j1: 0.000, j2: 0.000, j3: 0.000, j4: 0.000, j5: 0.000, j6: 0.000; x: 0.000, y: 0.000, z: 0.000; Roll: 0.000, Pitch: 0.000, Yaw: 0.000

tutorialApp
app.UIFigure
app.HomeButton
app.CommsLabel
app.MovePoseButton
app.YawEditField_2
app.PitchEditField_2
app.RollEditField_2
app.zEditField_2
app.yEditField_2
app.xEditField_2
app.OrientationradLabel_2
app.PositionmLabel_2
app.TargetStateLabel
app.CurrentStateLabel
app.YawEditField
app.PitchEditField
app.RollEditField

Name: tutorialApp
Version: 1.0

Fig. 7. Layout of Matlab controller for the Niryo Ned robot



```
        NedIPLabel              matlab.ui.control.Label
    end

    %% Matlab App for Niryo Ned Control by Praneel Chand
    %% Credits and thanks to Nicolas Guy, Niryo, & The Brainary
    properties (Access = private)
        %insert global variable names here
        ned % ned robot model from urdf file
        curJoints % current joint angles in radians (joints 1 to 6
        curPosition % current position coordintes (x, y, z)
        curOrient % current orientation in radians (roll, pitch, yaw)
        learnCmd % learning mode command
        jointStates %subscription to read joint states topic (joint angles)
        robotStates %subscription to read robot states topic (position, rpy, quaternion)
        nedState % subscription to read ned states topic
        nedCmd % publish to ned commands topic
        learnClient % learning mode client
        ikClient % inverse kinematics client
    end

    methods (Access = private)

    end


    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
            %clear all
```

Fig. 8. Code view of Matlab controller for the Niryo Ned robot.

TABLE II. COMPARISON OF MOVEMENT WITH NIRYO STUDIO AND MATLAB CONTROLLER

| Position Name | Target Coordinates (x,y,z) (m) | Niryo Studio Coordinates (x,y,z) (m) | Matlab App Coordinates (x,y,z) (m) |
|---|---|---|---|
| Edge 1 (Origin) | (0.420, -0.095, 0.106) | (0.420, -0.095, 0.106) | (0.419, -0.099, 0.107) |
| Edge 2 | (0.420, 0.100, 0.106) | (0.420, 0.100, 0.106) | (0.421, 0.095, 0.106) |
| Edge 3 | (0.225, 0.100, 0.106) | (0.225, 0.100, 0.106) | (0.225, 0.097, 0.107) |
| Edge 4 | (0.225, -0.095, 0.106) | (0.225, -0.095, 0.106) | (0.223, -0.096, 0.106) |

## III. RESULTS AND DISCUSSION

### A. Position Tests

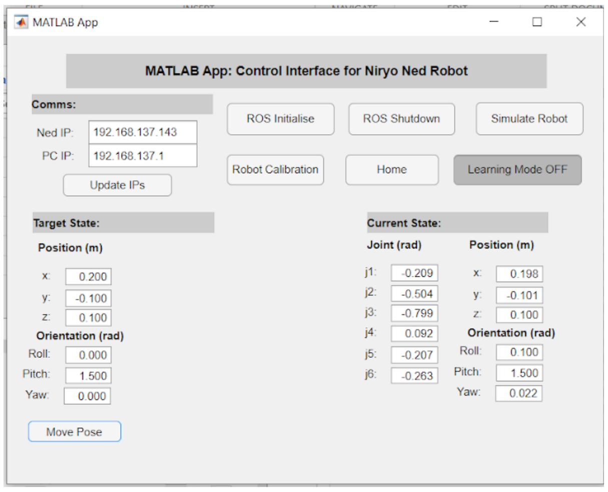
The functionality of the developed Matlab controller has been tested by sending commands to navigate the robot's tool center point (TCP) position from an initial home position [(x, y, z) = (0.135, 0.000, 0.213)] to the four edge markers in the predefined workspace. Table II details the target positions and positions achieved by Niryo Studio and the Matlab App. Navigation to each position was repeated 10 times. There was no variance in the coordinates reached by Niryo Studio and Matlab App over the 10 repetitions. Therefore, repeatability is identical for both software. However, the accuracy of Niryo Studio is better with zero error between the target and achieved coordinates. There is a mean absolute error (MAE) of 0.0010 m, 0.0033 m, and 0.0005 m, in the x, y, and z axes, respectively for the Matlab App. These errors are small in comparison with the gripper claws opening range of 0.0238 m. However, it will be useful to further investigate the ROS service and topic communication between the computer and the Ned robot to possibly improve accuracy. Fig. 9 and Fig. 10 show screenshots of the Matlab controller being used with the physical robot.

The overhead camera has been interfaced to the developed Matlab controller. Initial results on the using the Matlab controller to recognize and locate markers in the workspace is shown in Fig. 11. More details about object detection and recognition are available in [20] and [21].

### B. Video Demonstrations

Two short demonstration videos that verify the control of the robot through the Matlab App are also available online:

- Video 1 – this video shows basic communication between the Matlab App and the Ned robot. It also shows a comparison with Niryo Studio.
- Video 2 – this video shows the Ned robot arm navigating the four edge markers in the predefined workspace.



Comms:
Ned IP: 192.168.137.143
PC IP: 192.168.137.1

Target State:
Position (m)
x: 0.200
y: -0.100
z: 0.100
Orientation (rad)
Roll: 0.000
Pitch: 1.500
Yaw: 0.000

Current State:
Joint (rad)
j1: -0.209
j2: -0.504
j3: -0.799
j4: 0.092
j5: -0.207
j6: -0.263
Position (m)
x: 0.198
y: -0.101
z: 0.100
Orientation (rad)
Roll: 0.100
Pitch: 1.500
Yaw: 0.022

Fig. 9. Matlab controller in operation.

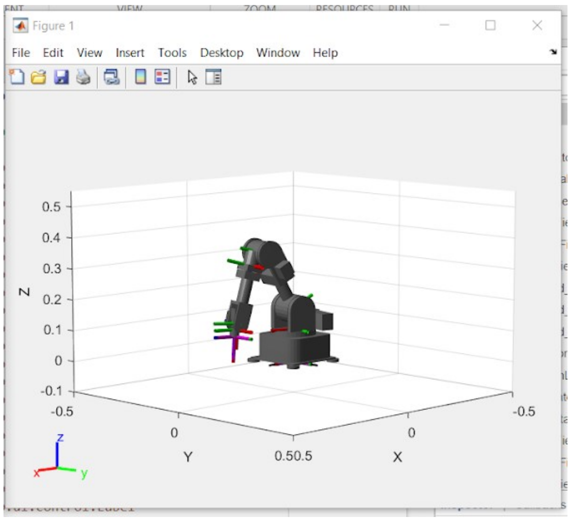
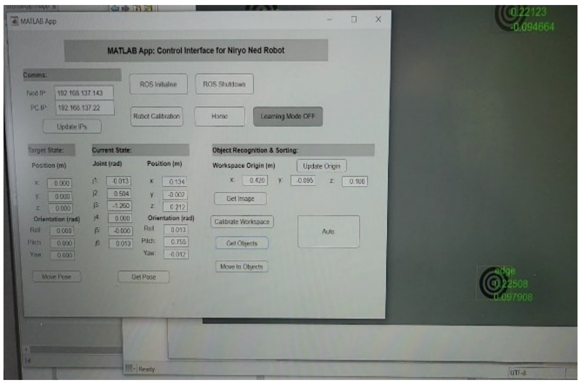


Fig. 10. Matlab controller robot simulation.



Comms:
Ned IP: 192.168.137.143
PC IP: 192.168.137.22

Target State:
Position (m)
x: 0.000
y: 0.000
z: 0.000
Orientation (rad)
Roll: 0.000
Pitch: 0.000
Yaw: 0.000

Current State:
Joint (rad)
j1: -0.013
j2: 0.504
j3: -1.260
j4: 0.000
j5: -0.000
j6: 0.013
Position (m)
x: 0.134
y: -0.002
z: 0.212
Orientation (rad)
Roll: 0.013
Pitch: 0.755
Yaw: -0.012

Object Recognition & Sorting:
Workspace Origin (m)
x: 0.420 y: -0.095 z: 0.108

Fig. 11. Matlab controller with object recognition.

## IV. Conclusion

This paper has presented the design and development of a Matlab controller for the Niryo Ned robot. The controller is dependent on setting up a wireless communication link between the physical robot and a Windows computer running Matlab. The developed controller will be coupled with a vision system and machine learning algorithms to implement object sorting/handling systems.

## Acknowledgment

The author greatly appreciates the assistance of Niryo and The Brainary for their technical support.

## References

[1] A. Dzedzickis, J. Subačiūtė-Žemaitienė, E. Šutinys, U. Samukaitė-Bubnienė, and V. Bučinskas, "Advanced Applications of Industrial Robotics: New Trends and Possibilities," *Applied Sciences,* vol. 12, no. 1, p. 135, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/1/135.

[2] F. Negrello, H. S. Stuart, and M. G. Catalano, "Hands in the Real World," (in English), *Frontiers in Robotics and AI,* Review vol. 6, 2020-January-30 2020, doi: 10.3389/frobt.2019.00147.

[3] M. Hernández-Ordoñez, M. A. Nuño-Maganda, C. A. Calles-Arriaga, O. Montaño-Rivas, and K. E. Bautista Hernández, "An Education Application for Teaching Robot Arm Manipulator Concepts Using Augmented Reality," *Mobile Information Systems,* vol. 2018, p. 6047034, 2018/08/06 2018, doi: 10.1155/2018/6047034.

[4] L. Chu, Y.-L. Ting, and Y. Tai, "Building STEM Capability in a Robotic Arm Educational Competition," presented at the Learning and Collaboration Technologies. Human and Technology Ecosystems: 7th International Conference, LCT 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part II, Copenhagen, Denmark, 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-50506-6_28.

[5] P. Chand, K. Kumar, and K. Kumar, "Development of an Intelligent Control Based Vehicle Following System," in *Mobile Robotics: Principles, Techniques, and Applications*, A. Montgomery Ed. New York, USA: Nova Science Publishers, 2016, ch. IV, p. 127.

[6] M. A. Peshkin, J. E. Colgate, W. Wannasuphoprasit, C. A. Moore, R. B. Gillespie, and P. Akella, "Cobot architecture," *IEEE Transactions on Robotics and Automation,* vol. 17, no. 4, pp. 377-390, 2001, doi: 10.1109/70.954751.

[7] E. Demaitre. "Robot Prices Drop as AI Investments, Ethics Worries Increase, Finds Stanford HAI AI Index 2022 Report." Robotics247. https://www.robotics247.com/article/robot_prices_drop_ai_investments_ethics_worries_increase_finds_stanford_hai_ai_index_2022_report (accessed 4 July 2022, 2022).

[8] ABB. "IRB 120 For flexible and compact production." ABB. https://new.abb.com/products/robotics/industrial-robots/irb-120 (accessed 4 July 2022, 2022).

[9] Omron. "17201-36000 Viper 650." Omron. https://store.omron.co.nz/product/r6a-0018a (accessed 4 July 2022, 2022).

[10] Kuka. "KR 4 Agilus." https://www.kuka.com/en-de/products/robot-systems/industrial-robots/kr-4-agilus (accessed 4 July 2022, 2022).

[11] Universal-Robots. "Universal Robot UR3e." Universal Robots. https://www.universal-robots.com/products/ur3-robot/ (accessed 4 July 2022, 2022).

[12] Niryo. "NED User Manual." Niryo. https://docs.niryo.com/product/ned/v4.0.0/en/index.html (accessed 30 June 2022, 2022).

[13] WLKATA. "WLKATA Mirobot Education Kit." https://www.wlkata.com/product/wlkata-mirobot-education-kit (accessed 4 July 2022, 2022).

[14] Elephant-Robotics. "MyCobot 280 Pi." Elephant Robotics. https://shop.elephantrobotics.com/en-nz/collections/mycobot/products/mycobot-pi-worlds-smallest-and-lightest-six-axis-collaborative-robot?variant=39585945649238 (accessed 4 July 2022, 2022).

[15] Niryo. "Niryo Studio - Robot connection." Niryo. https://docs.niryo.com/product/niryo-studio/v4.0.1/en/source/connection.html (accessed 5 July 2022, 2022).

[16] Microsoft. "Use your Windows PC as a mobile hotspot." Microsoft. https://support.microsoft.com/en-us/windows/use-your-windows-pc-as-a-mobile-hotspot-c89b0fad-72d5-41e8-f7ea-406ad9036b85#WindowsVersion=Windows_10 (accessed 5 July 2022, 2022).

[17] Niryo. "Setup Matlab and the ROS Toolbox for Ned." https://docs.niryo.com/applications/ned/v1.0.0/en/source/tutorials/setup_ned_matlab_ros_toolbox.html (accessed 5 July 2022, 2022).

[18] Mathworks. "ROS System Requirements." Mathworks. https://au.mathworks.com/help/ros/gs/ros-system-requirements.html#responsive_offcanvas (accessed 5 July 2022, 2022).

[19] Niryo. "Publish and Subscribe on Ned with Matlab ROS Toolbox." Niryo. https://docs.niryo.com/applications/ned/v1.0.0/en/source/tutorials/control_ned_matlab_ros_publish_subscribe.html (accessed 5 July 2022, 2022).

[20] R. Kumar, S. Lal, S. Kumar, and P. Chand, "Object detection and recognition for a pick and place Robot," in *Asia-Pacific World Congress on Computer Science and Engineering*, 4-5 Nov. 2014 2014, pp. 1-7, doi: 10.1109/APWCCSE.2014.7053853.

[21] P. Chand, "Investigating Vision Based Sorting of Used Items," in *2022 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, 13-15 Sep. 2022 2022, pp. 1-5.